Санкт-Петербургский государственный университет
Прикладная математика и информатика
Исследование операций и принятие решений в задачах оптимизации, управления и экономики

Арефьев Сергей Александрович

Модель оценивания пар игроков в теннис

Выпускная квалификационная работа

Научный руководитель:
к. ф.-м. н., доцент В. В. Бухвалова

Рецензент:
Старший преподаватель
А. С. Зациорский

Санкт-Петербург
2017

Saint Petersburg State University
Applied Mathematics and Computer Science
Operation Research and Decision Making in Optimisation, Control and Economics Problems

Arefyev Sergey

Model of tennis-pair valuation

Bachelor's Thesis

Scientific Supervisor:
Assistant professor Vera Buhvalova

Reviewer:
Senior Lecturer Artem Zaciorsky

Saint Petersburg
2017

Оглавление

Введение

Большой теннис — один из самых популярных видов спорта в мире. Ежегодно серия ATP (Ассоциация теннисистов-профессионалов) проводит более 6000 матчей. Трансляции финалов крупнейших турниров наблюдают десятки миллионов человек. Из-за огромного интереса к этому виду спорта, сумма ставок растет быстрыми темпами. Всплеск в области изучения методов машинного обучения и большое количество информации о матчах обусловили развитие построения моделей для оценивания теннисистов.

Самые первые модели [1] для оценки результатов теннисных матчей строились лишь на основе вероятности выигрыша очка теннисистом. Такие модели просты для понимания, но не учитывают множество факторов (усталость, рейтинг, стиль игрока и т.д.). A. Madurska [8] рассмотрел модели с использованием метода общих оппонентов. Данный метод улучшил результаты предыдущих работ.

С развитием машинного обучения стали использоваться такие алгоритмы, как логистическая регрессия и нейронные сети. Стал увеличиваться набор характеристик игроков и матча, используемых в алгоритмах. Данная работа отталкивается от результатов, полученных в работе M. Sipko [9]. В отличии от предложенной там модели, были применены новые алгоритмы машинного обучения, которые хорошо показывают себя на практике в задачах классификации. Метод создания новых характеристик игроков был изменен.

Работа состоит из 6 глав и введения. В 1 главе ставится постановка задачи и даются основные определения. Глава 2 содержит описание используемых алгоритмов машинного обучения. В главе 3 рассказывается о необходимых преобразование данных для модели. Глава 4 содержит описание обучения модели. В главе 5 показаны результаты работы модели и её

реализация. В главе 6 описаны итоги работы. Приложение содержит в себе правила игры в теннис.

1. Постановка задачи и основные определения

Имеется база данных матчей в серии ATP с 2003 по 2016 гг. В исходной базе каждый матч имеет 50 характеристик: данные об игроках, результат матча, специфические характеристики матча и игроков.

Используя базу, разработать модель оценивания результатов будущих матчей, протестировать её на матчах 2016 года. Для оценивание модели используется логарифмическая функция потерь (см. раздел 5.1).

1.1. Ставки в теннисе

За несколько часов до начала матча, букмекерской конторы показывают свои коэффициенты на победу для каждого из игроков. Интересно посчитать возможную прибыль. Для этого необходимо иметь стратегию, которая решает необходимо ли ставить на данный матч.

Рассмотрим критерий Келли [6].

Определение 1.1. *Пусть имеется капитал размером B, коэффициент букмекера равен K и оценка игрока равна V. Тогда по* ***Критерию Келли*** *оптимальная ставка S равняется:*

$$S = \begin{cases} B\left(\frac{KV-1}{K-1}\right), & V \geq 0.5,\ V \geq 1/K, \\ 0, & \textit{иначе.} \end{cases}$$

Для избежания быстрого расхода капитала, максимальная ставка ограничивается 10 процентами от размера капитала.

1.2. База данных

Для построения модели использовалась база данных **OnCourt**. База данных ежедневно обновляется и имеет большой набор данных, доступных открыто в интернете. Ниже представлены характеристики, используемые для построения модели:

Таблица 1. База данных: группы характеристик и их состав

Общие характеристики игроков
Имя игрока
Вес*
Рост*
Место в рейтинге ATP
Рейтинг ATP
Характеристики турнира
Название турнира
Тип покрытия
Счет матча
Дата матча
Характеристики игроков в матче (C1)
Количество эйсов
Количество двойных ошибок
Количество невынужденных ошибок*
Количество виннеров*
Максимальная скорость подачи*
Средняя скорость первой подачи*
Средняя скорость второй подачи*
Характеристики игроков в матче (C2)
Количество поданных первых подач, Количество попыток подать первых подач
Количество выигранных очков на первой подаче, Количество разыгранных очков на первой подаче
Количество выигранных очков на второй подаче, Количество разыгранных очков на второй подаче
Количество выигранных очков на своей подаче, Количество разыгранных очков на своей подаче
Количество выигранных очков на первой подаче соперника, Количество разыгранных очков на первой подаче соперника
Количество выигранных очков на второй подаче соперника, Количество разыгранных очков на второй подаче соперника
Количество выигранных очков на чужой подаче, Количество разыгранных очков на чужой подаче
Количество удачных выходов к сетке*, Количество выходов к сетке*
Количество выигранных брейк поинтов, Количество брейк поинтов
Количество спасенных брейк поинтов, Количество брейк поинтов соперника
Количество выигранных очков, Количество разыгранных очков
Количество выигранных геймов, Количество разыгранных геймов
Количество побед в личных встречах, Количество личных встреч

Характеристики, помеченные *, могут иметь пропущенные значения.

2. Алгоритмы машинного обучения

Машинное обучение — раздел искусственного интеллекта, изучающий алгоритмы, способные обучаться на основании некоторой выборки данных. Машинное обучение с учителем используется для построения функции из набора пар, состоящих из вектора, полученного на входе и необходимого значения на выходе. В этой главе приведено описания алгоритмов, которые используются для обучения модели:

1. Gradient Boosting
2. Random forest
3. Нейронные сети

2.1. Задача классификации

Имеется матрица объектов X, состоящая из N наблюдений x_i, каждое из которых является вектором размера M(количество признаков):

$$X = (x_i)_{i=1}^{N}, \quad x_i \in \mathbb{R}^M,$$

и вектор результатов

$$y = (y_i)_{i=1}^{N}, \quad y_i \in \{0, 1\}.$$

Каждому наблюдению x_i соответствует результат y_i. Предполагается, что в векторе наблюдений x_i не содержится информации, доступной после-начала матча.

$$y_i = \begin{cases} 1, \text{ победа первого игрока в паре,} \\ 0, \text{ победа второго игрока в паре.} \end{cases}$$

2.2. Gradient Boosting

Данный раздел основан на переводе статьи [4].

Рассмотрим алгоритм Gradient Boosting для задачи классификации. Основная идея данного алгоритма заключается в последовательном обучении базовых алгоритмов (в данном случае деревьев решений), каждый из которых улучшает результат основного алгоритма.

Дерево решений представимо в виде:

$$\mathcal{F} = \{f(x) = w_q(x)\}\ (q : \mathbb{R}^M \to T, w \in \mathbb{R}^t), \tag{1}$$

где w — вектор значений в листьях, q — функция, отображающая набор признаков в определенный лист дерева, T — количество листьев.

Для объекта $x_i \in X$ предсказание выражается в виде:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i),\ f_k \in \mathcal{F}. \tag{2}$$

При построении ансамбля деревьев на шаге t жадно добавляется дерево, которое в композиции с предыдущими деревьями наилучшим образом минимизирует следующую функцию:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{N} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)) + \theta(f_t), \tag{3}$$

где $\theta(f) = \gamma T + 1/2\lambda||w||^2$ — терм регуляризации, γ и λ — параметры регуляризации для контроля переобучения.

Для нахождения данного дерева, раскладываем функцию в ряд Тейлора до второго порядка и после преобразований получаем:

$$\hat{\mathcal{L}}^{(t)} = \sum_{j=1}^{T} G_{I_j} w_j + \frac{1}{2}(H_{I_j} + \lambda) w_j^2) + \gamma T,$$
$$H_{I_j} = \sum_{i \in I_j} \partial^2_{\hat{y_i}^{(t-1)}} l(y_i, \hat{y_i}^{(t-1)}), \tag{4}$$
$$G_{I_j} = \sum_{i \in I_j} \partial_{\hat{y_i}^{(t-1)}} l(y_i, \hat{y_i}^{(t-1)}).$$

После этого преобразования, для структуры дерева $q(x)$ можно найти наилучшие веса $w_j^* = -\frac{G_{I_j}}{H_{I_j} - \lambda}$ и значение функции

$$\hat{\mathcal{L}}^{(t)} = -\frac{1}{2} \sum_{j=1}^{T} \frac{(G_{I_j})^2}{H_{I_j} - \lambda} + \gamma T.$$

Далее строится дерево, жадно разбивая лист на ветви. Качество разделения находится с помощью формулы:

$$\mathcal{L}_{split} = \frac{1}{2}\left(\frac{G_{I_l}^2}{H_{I_l} + \lambda} + \frac{G_{I_r}^2}{H_{I_r} + \lambda} - \frac{(G_{I_l} + G_{I_r})^2}{H_{I_l} + H_{I_r} + \lambda}\right) - \gamma, \tag{5}$$

2.3. Random Forest

Алгоритм Random Forest [3], в отличие от метода Gradient Boosting, строит деревья решений независимо друг от друга. Опишем процедуру построения алгоритма:

1. Из X выбирается случайная выборка X' размера N с повторениями.
2. Берется $m \in M$ случайных признаков и на них строится решающее дерево.
3. Дерево строится до того момента, пока в листьях не останутся представители одного класса.
4. После построений необходимого количества деревьев, результатом алгоритма будет усреднение их ответов.

2.4. Нейронная сеть

В модели используется нейронная сеть, обученная с помощью метода обратного распространения ошибки [5].

Нейронная сеть состоит из последовательности слоев, каждый из которых включает в себя набор нейронов. Нейрон одного слоя имеет связи со всеми нейронами предыдущего слоя, каждая из которых имеет свое вес. Таким образом имеется взвешенный ориентированный граф, вершинами которого являются нейроны, а ребрами — связи с весами. Выделим 3 типа слоев:

1. входной слой (имеет M нейронов, где M — количество признаков),
2. скрытый слой,
3. выходной слой (имеет 1 нейрон).

Во входной слой подается векторы из выборки X, значения нейронов скрытого и выходного слоя равны:

$$s = f(\sum_{i=1}^{n} w_i * x_i),$$

где n_i — значение нейрона i предыдущего слоя, w_i — вес связи между нейроном s и i, n — количество нейронов на предыдущем слое,
$f : \mathbb{R}^n \to \mathbb{R}$ — функция активации.

В модели были использованы две функции активации:

- Relu: $f(x) = \max(x, 0)$,
- Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$.

Теперь опишем алгоритм обратного распространения ошибки:

1. Генерация небольших случайных значений для весов связей.
2. Подача выборки X на входной слой.
3. Вычисление выхода сети.
4. Вычисление разницы между выходом сети и значением вектора результатов y.
5. Корректировка весов для минимизации функции потери нейронной сети.
6. Повторять шаги с 2 до 5, пока значение функции потери не станет приемлемым.

Дла корректировки весов, в шаге 5, используется алгоритма Adam [7].

3. Преобразование данных

Для обучения модели необходимо на начальных данных построить матрицу X объектов (матчей), в которой M строк и N столбцов. Каждой строке X соответствует 1 матч. Предполагается, что в ней не содержаться информации, доступной после начала игры. Каждому столбцу матрицы A соответствует характеристика игроков или матча. Имеется вектор y размера M, состоящий из 1 или 0 (победа первого или второго игрока).

Разделы 3.1, 3.2, 3.5, 3.6 и 3.8 базируются на работе [9].

3.1. Веса покрытий

Для игры в теннис используется 4 основных типа покрытия (трава, грунт, хард, ковровое покрытие). В зависимости от стиля, теннисисты предпочитают различные типы покрытий. За веса покрытий возьмем коэффициенты корреляции между успешностью игроков на разных покрытиях. Для каждой пары покрытий (a, b), он считается следующим образом:

$$WC(a,b) = \frac{\sum_{i=1}^{g}(a_i - \hat{a})(b_i - \hat{b})}{(g-1)s_a s_b}, \tag{6}$$

- a_i — процент побед игроком i на покрытии a,
- b_i — процент побед игроком i на покрытии b,
- s_a — стандартное отклонение в проценте побед на покрытии a,
- s_b — стандартное отклонение в проценте побед на покрытии b,
- $\hat{a}$ — средний процент побед на покрытии a,
- $\hat{b}$ — средний процент побед на покрытии b,
- g — количество игроков.

Таблица 2. Таблица с коэффициентами корреляций

	Хард	**Грунт**	**Ковровое**	**Трава**
Хард	1			
Грунт	0.28	1		
Ковровое	0.35	0.31	1	
Трава	0.24	0.14	0.25	1

3.2. Вес по времени

Для вычисления веса по времени используется 2 формулы.

$$WT_1(t) = \min(f^t, f), \tag{7}$$

где $f = 0.8$ (значение взято из работы [9]), t — временной интервал между датой матча и моментом анализа (в годах).

$$WT_2(t) = f^t, \tag{8}$$

где $f = 0.26$ (данное значение подобрано из соображений, что будут выбраны матчи минимум за год и максимум за 2,5 года).

3.3. Усредненные характеристики

В работе предложено изменить усредненные характеристики, предложенные в работе [9], добавив параметр t (временной интервал).

Для построения новых характеристик используются наборы $C1$ и $C2$ из таблицы (1).

Усредненная характеристика h из группы $C1$ вычисляется для игрока p, временного интервала t и вектора значений характеристик $R = (r_{p1}, r_{p2}, \ldots, r_{pk})$ по формуле

$$h(w, t, p) = \sum_{i=1}^{k} w_i r_{pi}, \tag{9}$$

где w_i — вес матча i.

Усредненная характеристика h из группы $C2$ вычисляется для игрока p, временного интервала t и пары векторов значений характеристик $R^1 = (r^1_{p1}, r^1_{p2}, \ldots, r^1_{pk}), R^2 = (r^2_{p1}, r^2_{p2}, \ldots, r^2_{pk})$ по формуле

$$h(w, t, p) = \sum_{i=1}^{k} \frac{w_i r^1_{pi}}{r^2_{pi}}, \tag{10}$$

где w_i — вес матча i.

Значение t равняется трем месяцем, шести месяцем, одному, двум годам. Вес равен единице. Также параметр t может не использоваться, тогда мы будем рассматривать все матчи игрока p за его карьеру. В этом случае веса равны еденице, произведению веса покрытия (6) и веса по времени (7) или произведению (6) и (8).

3.4. Усредненные характеристики и тип корта

Так как игроки показывают различные результаты на разных типах кортов, то логично брать значения характеристики среди матчей того же типа корта. Тогда можно изменить формулу (9) добавив параметр ct (тип корта).

$$h(w, t, p, ct) = \sum_{i=1}^{k} w_i r_{pi}, \tag{11}$$

где w_i — вес матча i, ct — тип корта, для которого берутся характеристики.

Значение t равняется шести месяцем, одному, двум годам. Вес равен единице.

3.5. Метод общих оппонентов

Рассмотрим характеристику игрока h из набора $C1$ для пары игроков p_1, p_2. Для построения новой усредненной характеристики с помощью метода общих оппонентов $hc(w, t, p_1)$ необходимо найти вектор игроков $P = (po_1, po_2, ..., po_k)$ с которыми играли p_1 и p_2 за промежуток времени равный t. Далее для каждого игрока $po_i \in P$ берем набор матчей сыгранных против p_1 и получаем вектор T_i^1 и для него считаем усредненную характеристику $h_i(w, t, p_1)$ по формуле (9). В итоге, для p_1 имеем вектор $R_1 = (h_{11}(w, t, p_1),\ h_{21}(w, t, p_1), ...,\ h_{k1}(w, t, p_1))$. Тогда значение усредненной характеристики, рассчитанной с помощью метода общих оппонентов для игрока p_1 равно

$$hc(w, t, p_1)_1 = \frac{\sum_{i=1}^{k} h_{i1}}{k},\ h_{i1} \in R_1. \tag{12}$$

Аналогично считается для характеристик из набора $C2$.

Значение t равняется одному или двум годам. Вес, для данного t, равные единице. Также параметр t, может не использоваться и тогда веса равняются единице, произведению веса покрытия (6) и веса по времени (7).

Для каждой пары игроков значения характеристик, полученных из формул (9) , (10), (11) и (12), представляются в виде их разности.

3.6. Степень неопределенности матча

После вычислений весов матча для каждой пары игроков p_1, p_2 находится степень неопределенности матча

$$u(p_1, p_2) = \sum_{i=1}^{k_1} w_{i1} \cdot \sum_{i=1}^{k_2} w_{i2}, \tag{13}$$

где w_{ij}, w_{i2}, веса матчей игроков p_1 и p_2, равные произведению (6) и(7), k_1 и k_2 количество матчей, сыгранных p_1 и p_2 за 2 года соответственно.

Данный показатель используется как характеристика для игроков. Неопределенности матча показывает насколько много статистики собрано для пары теннисистов. Поэтому, с помощью неё, можно отобрать матчи с наибольшем количеством статистики для более точной работы модели.

3.7. Стиль игроков

Каждый игрок имеет свой индивидуальный стиль игры, каждый из которых имеет свои преимущества и недостатки. В работе выделены 8 стилей игры в теннис для top100 игроков в рейтинге ATP:

- Пассивный игрок задней линии,
- Атакующий игрок задней линии,
- Игрок, играющий на задней линии,
- Игрок, играющий у сетки,
- Игрок, действующий по всему корту,
- Игрок, использующий слабые стороны игрока,
- Игрок, полагающийся на свою подачу,
- Игрок, полагающийся на невынужденные ошибки соперника.

Стили игроков в матрице объектов представлены вектором, у которого значение равно единице, если игрок имеет данный стиль и нулю, если не имеет. Добавим, что топовые игроки могут иметь в арсенале несколько стилей игры.

3.8. Усталость игрока

Большинство турниров в теннисе проходят за одну неделю. Для победы необходимо выиграть 5 игр. Поэтому большую роль играет усталость теннисиста. Для подсчета усталости мы возьмем формулу из работы [9]. В расчёт идут матчи, сыгранные за прошедшие 3 дня. Усталость для каждого игрока равна

$$g0.75^t,$$

где t — количество дней прошедших с матча, g — количество геймов в матче.

3.9. Работа с пропущенными данными

В начальной базе данных имеются пропущенные данные. Это физический характеристики игроков (рост, вес) и некоторые статистики игроков (помечены * в таблице 1). Также необходимо заполнить пропущенные значения стилей игроков, которые не входят в top100 рейтинга ATP. Заполняются пропущенные данные 2 способами:

1. Для физических характеристик — медианой характеристики
2. Для остальных характеристик — методом Random forest.

Для метода Random Forest использовались значения усредненных характеристик из набора C1 и C2 у которых не имеется пропущенных данных.

3.10. Итоговый набор характеристик

Итоговый набор характеристик, который используется для алгоритмов машинного обучения, разбит на 3 части:

1. Характеристики, взятые из начальной базы данных,
2. Усредненные характеристики набора C1 и C2,
3. Другие характеристики (см. раздел 4.5 — 4.7).

Таблица 3. Характеристики из начальной базы данных

Вес игрока
Рост игрока
Место игрока в рейтинге ATP
Рейтинг игрока ATP
Тип покрытия

Таблица 4. Другие характеристики

Степень неопределенности матча
Усталость игроков
Стиль игроков

Таблица 5. Усредненные характеристики набора C1 и C2

Набор временных промежутков (в г.)	**Использовался метод общих оппонентов**	**Вес**
[0.25, 0.5, 1, 2]	нет	1
[0.5, 1, 2] [a]	нет	1
–	нет	(6) * (7)
–	нет	(6) * (8) [b]
[1, 2]	да	1
–	да	(6) * (7)
–	да	(6) * (8)

[a] использовался метод из раздела 3.4

[b] рассмотриваются матчи, у которых вес больше, чем 0.027

В итоге получили 268 характеристик, которые будут использоваться алгоритмами машинного обучения.

4. Обучение модели

При использовании алгоритмов машинного обучения часто проявляется эффект переобучения, при котором ошибка обученного алгоритма на тестовой выборки получается существенно выше, чем ошибка на обучающей выборке. Поэтому, при обучении алгоритмов, нужно из обучающей выборки взять набор данных для проверки (валидации). Для обучения мы разбиваем наш временной интервал на 3 части:

1. Обучение алгоритмов проводится на данных с 2003 по 2014 гг,
2. Валидация алгоритмов проводится на данных 2015 г,
3. Тестовый набор взят на данных 2016. На этом наборе мы протестируем итоговую модель.

4.1. Оценка модели

Необходимо построить алгоритм $A : X \to y$, который для пары i оценивает вероятность победы первого игрока над вторым.

Результатом работы алгоритма является вектор $p = (0, 1)^N$. Для оценки качества алгоритмов используется логарифмическая функция потерь.

Определение 4.1. *Логарифмическая функция потерь (logloss):*

$$f(y, p) = \frac{1}{N} \sum_{i=1}^{N} l(y_i, p_i) \,,$$

где $l(y, p) = -y \ln p + (y - 1) \ln (1 - p)$, y *— результат матча,*
p *— вероятность победы первого игрока, вычисленная алгоритмом* A.

4.2. Отбор характеристик и настройка параметров

Основными параметрами модели являются:

1. Интервал начал интервалов обучения [2003, 2010],
2. Доля выбранных матчей, с использованием степени неопределённости [0.5, 0.4, 0.35, 0.3, 0.25, 0.2],
3. Набор характеристик, используемых для обучения (только для алгоритма Gradient Boosting).

Для выбора матчей, с помощью степени неопределенности, необходимо отсортировать матчи по этой характеристике. Далее берутся доля матчей, у которых значение степени неопределенности наибольшее.

Для алгоритмов Random Forest и нейронные сети были перебраны всевозможные комбинации параметров модели и выбраны те, на которых значение функции потерь минимально.

Наборы параметров для нейронных сетей:

Год начала обучения	**Доля выбранных матчей (ст. неопр.)**
2005	0.25
2006	0.3

Набор параметров для Random Forest:

Год начала обучения	**Доля выбранных матчей (ст. неопр.)**
2008	0.3

Алгоритм Gradient Boosting позволяет получить список важности характеристик. Поэтому для каждой комбинации параметров моделей характеристики были отсортированы по показателю важности и последовательно удалялись характеристики с наименьшим показателем и обучался алгоритм. После этого было выбрано три различных наборов параметров модели для Gradient Boosting, на которых значение функции потерь минимально:

Год начала обучения	**Доля выбр. матчей (ст. неопр.)**	**Количество отобранных характеристик**
2007	0.25	98
2007	0.3	126
2006	0.3	68

4.3. Настройка параметров алгоритмов

Следующим этапом в в создании модели является подбор параметров алгоритмов. Для алгоритмов Gradient Boosting и Random Forest списки параметров, среди которых осуществлялся перебор, и списки параметров, которые были выбраны, выглядят следующем образом:

Алгоритм	**Gradient Boosting**	
Название параметров	**Набор параметров**	**Выбранный параметр**
max_depth	[3, 4, 5, 6, 7]	3
min_child_weight	[1, 2, 3, 4, 5]	1
learning_rate	[0.1, 0.05]	0.1
gamma	[0, 0.1, 0.5, 1, 5, 10]	0
subsample	[0.5, 0.6, 0.7, 0.8, 0.9, 1]	1
colsample_bytree	[0.5, 0.6, 0.7, 0.8, 0.9, 1]	1
n_estimators	[70, 100, 130, 150, 200]	100

Алгоритм	**Random Forest**	
Название параметров	**Набор параметров**	**Выбранный параметр**
max_features	[sqrt, log2, None, 0.1]	None
min_samples_leaf	[20, 30, 40, 50, 70, 100]	20
n_estimators	[10, 15, 20, 30, 50]	100

Нейронные сети имеют большое количество параметров, поэтому была выбрана архитектура нейронной сети, выглядевшая следующем образом:

Слой	**Количество нейронов**	**Функция активации**	Dropout
Входной	268	Relu	—
Скрытый 1	**nn**	Relu	**dr**
Скрытый 2	**nn***0.5	Relu	**dr** - 0.1
Выходной	1	Sigmoid	—

Размер входного слоя равен количеству полученных признаков.

Для настройки параметров нейронной сети необходимо выбрать значения **nn** и **dr**. Список параметров, среди которых осуществлялся перебор, и список параметров, которые были выбраны, выглядят следующем образом:

Алгоритм	**Нейронная сеть**	
Название параметров	**Набор параметров**	**Выбранный параметр**
nn	[50, 100, 150, 200]	150
dr	[0.2, 0.3, 0.4, 0.5]	0.4

4.4. Ансамбль алгоритмов

Последним этапом будет построение ансамбля алгоритмов. Для этого необходимо для каждой пары игроков взять среднее значение от оценок алгоритмов.

Такой метод называется Bagging [2], и в большинстве случаев работает лучше, чем оценки каждого из алгоритмов.

5. Результаты и реализация

5.1. Результат работы модели

Оценим модель с помощью логарифмической функции потери. Тестирование проводилось на данных 2016 года. Ниже приведены оценки результатов алгоритмов и их ансамбля, а также результат работы M.Sipko. Напомним, что чем значение *logloss* (4.1) меньше, тем точнее алгоритм.

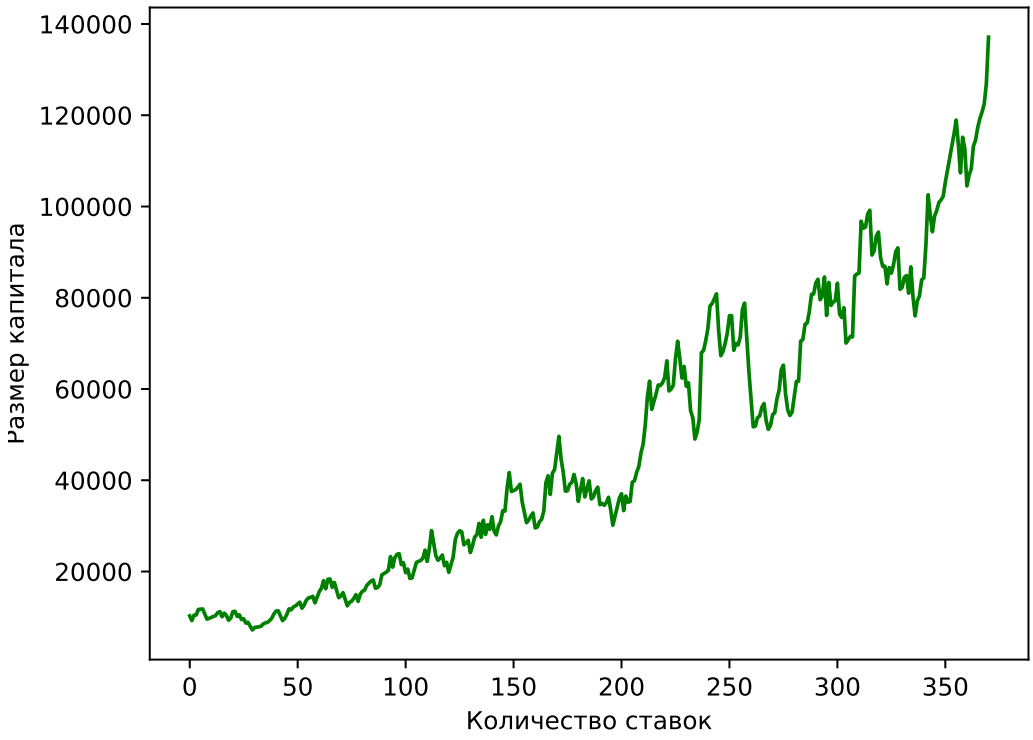
Алгоритмы	Значение *logloss*
Программа M.Sipko	**0.6111**
Gradient Boosting 1	0.5738
Gradient Boosting 2	**0.5726**
Gradient Boosting 3	0.5754
Random forest 1	**0.5826**
Neural network 1	**0.5836**
Neural network 2	0.5837
Ансамбль алгоритмов	**0.5691**

5.2. Симуляция ставок в букмекерской конторе

Ставки ставятся, с использованием критерий Келли (1.1). Для симуляции был выбран 2016 год. Коэффициенты взяты у конторы Betfair.

Начальный капитал составляет \$10 000. По критерию Келли смоделировано 371 ставка и на конец года он составлял \$137 000, таким образом капитал за год вырос в 13,7 раз. Для сравнения, модель M.Sipko за год увеличили капитал в 6 раз.

Ниже приведён график зависимости размера банка от количества поставленных ставок:



5.3. Программная реализация

Программа написана на языке **Python 3.5**. Использовались библиотеки: **XGBoost** (для алгоритма Gradient Boosting), **Scikit-learn** (для алгоритма Random forest), **Keras** (для нейронных сетей), **Pandas, Numpy** (для работы с базой данных).

Текст программы размещен по адресу:
`https://yadi.sk/d/pstEP6aB3JRUdY/2017`.

Схема использования программы

1. Первичная загрузка базы данных (бд OnCourt является коммерческим продуктом и покупается на сайте `www.oncourt.info`).

2. Первичная обработка базы данных для создания модели. Этот процесс занимает около 16 часов.

3. Обновление базы данных и добавление оценок для новых матчей рекомендуется проводить каждый день. Оно занимает около 5 минут.

После обновления базы данных мы получаем оценки модели для каждого из двух игроков и размер ставки по критерию Келли (1.1), в зависимости от капитала и коэффициента букмекерской конторы.

6. Заключение

В ходе работы получены следующие результаты:

- разработана и протестирована модель для оценки пар игроков в мужском теннисе,
- описаны новые способы получения характеристик игроков,
- протестированы новые алгоритмы машинного обучения,
- результаты, полученные с помощью предложенной в работе модели, в среднем превосходят результаты из работы [9] на 7 процентов.

7. Приложение. Правила игры в теннис

Теннис — вид спорта с ракеткой и мячом, в который играют 2 или 4 человека. В нашей модели мы будем рассматривать одиночные матчи, в которые играют 2 теннисиста. Корт для тенниса имеет прямоугольную форму, посередине разделенную сеткой. Игроки располагаются с разных сторон сетки. Цель игры состоит в том, чтобы перебросить мяч на половину противника, а противник не смог перевести его на вашу половину корта. Вначале игры один из игроков назначается подающим. Если игрок за 2 попытки не смог перевести мяч на противоположную сторону, то соперник зарабатывает очко. Очки с теннисе начисляются следующим образом: 0, 15, 30, 40. Для победы в гейме надо набрать

минимум 4 очка и иметь на 2 очка больше чем соперник. Гейм не заканчивается, пока такая ситуация не произойдет. После каждого гейма игроки меняются подачами. Для победы в сете игрок должен выиграть минимум 6 геймов и иметь на 2 гейма больше чем у соперника. Если счет в сете становится равным 6:6 начинается тайбрейк. Для победы в нем надо набрать минимум 7 очков и на 2 больше, чем у соперника. Победителем в игре становится теннисист выигравший 2 или 3 сета (в зависимости от турнира).

Перечислим основные термины, используемые при описании хода матча:

- Двойная ошибка: игрок 2 раза ошибся при подаче — соперник получает очко.
- Эйс: игрок правильно исполнил подачу, а соперник не смог коснуться мяча.
- Виннер: активно выигранный мяч, который соперник не смог отбить.

- Невынужденная ошибка: во время розыгрыша, игрок не смог перевести мяч на чужую на половину корта.

- Брейк: ситуация, когда при выигрыше следующего мяча игрок выигрывает гейм на чужой подаче.

Список литературы

1. T. Barnett and S. R. Clarke. Combining player statistics to predict outcomes of tennis matches. *IMA Journal of Management Mathematics*, 16:113 – 120, 4 2005.
2. L. Breiman. Bagging predictors. *Machine Learning*, 24:123 – 140, 8 1996.
3. L. Breiman. Random forests. *Machine Learning*, 45:5 – 32, 10 2001.
4. T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. University of Washington, 2016.
5. G. Hinton D. Rumelhart and R. Williams. Learning representations by back-propagating errors. *Letters to nature*, 323:553 – 536, 10 1986.
6. J. Kelly. A new interpretation of information rate. *The Bell System Technical Journal*, 35:917 – 926, 7 1956.
7. D. Kingma and J. Ba. Adam: A method for stochastic optimization. Conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.
8. A. Madurska. A set-by-set analysis method for predicting the outcome of professional singles tennis matches. Technical report, Imperial College London, 2012.
9. M. Sipko. Machine learning for the prediction of professional tennis matches. Imperial College London. Final year project, 2015.